KYMENLAAKSON AMMATTIKORKEAKOULU

Tietotekniikka / Ohjelmistotekniikka

Tatu Mäkinen

PC-PELI UNITYLLÄ ANDROID-LAITTEILLE

Opinnäytetyö 2014

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

MÄKINEN, TATU	PC-peli Unityllä Android-laitteille
Opinnäytetyö	31 sivua
Työn ohjaaja	Paula Posio
Toimeksiantaja	Kotka Games
Huhtikuu 2014	
Avainsanat	peli, Unity, C#, JavaScript, Android, ohjelmointi

Pelien suosio on kasvanut merkittävästi viime vuosien aikana. Unity on yksi monista peliohjelmointiin tarkoitetuista työkaluista. Suosituimmat tämänhetkiset pelit on julkaistu mobiilialustoille. Suurin osa latauksista kuuluu Android-laitteille.

Opinnäytetyön tarkoituksena oli tehdä paikalliselle peliyritykselle Kotka Gamesille toimiva peli Unityllä. Pelin tuli toimia sekä PC:llä että Android-laitteilla. Pelin tekeminen aloitettiin elokuussa 2013 ja saatiin päätökseen saman vuoden lokakuun aikana.

Opinnäytetyön peli tehtiin käyttäen Unity3D-peliohjelmistoympäristöä sekä sen liitännäistä NGUI:ta. Ohjelmointikielenä käytettiin C#:a sekä osittain JavaScriptiä. Opinnäytetyössä esiteltiin Unity kertomalla sen komponenteista ja keskeisistä työssä käytetyistä metodeista. Sitten kerrottiin NGUI:n tuomista lisäominaisuuksista, jonka jälkeen kerrottiin lyhyesti Android-laitteista. Näiden asioiden jälkeen kerrottiin itse pelistä ja käytiin tekemisen työvaiheet läpi. Lopuksi oli vuorossa kirjoittajan omaa pohdintaa työn tekemisestä ja ongelmakohdista.

Opinnäytetyön aikana näki NGUI:n hyödyllisyyden mobiilipeleille ja -sovelluksille. Se säästää aikaa graafisen asettelunsa vuoksi, joka mahdollistaa myös kuvien korvaamisen uusilla. Yritys voi helposti hyödyntää tätä ominaisuutta ja tehdä pelistä omanlaisensa omalla grafiikallaan.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

MÄKINEN, TATU	PC-game to Android using Unity
Bachelor's Thesis	31 pages
Supervisor	Paula Posio
Commissioned by	Kotka Games
April 2014	
Keywords	game, Unity, C#, JavaScript, Android, programming

The popularity of games has increased tremendously over the last few years. Unity is one of many tools meant for game programming. The most popular games nowadays have been published for mobile platforms. Most of the downloads belong to Android devices.

The task was to create a working game for a local game company Kotka Games using Unity. The game had to be able to be played on both PC and Android. Making of the game started in August 2013 and finished during October of the same year.

The game was done by using the game development platform Unity3D and its plugin called NGUI. The used programming languages were C# and partly JavaScript. In this document Unity was introduced by telling about its components and the methods that were used in the project. Next the features of NGUI were explained and after that there was brief information about Android devices. After those topics the game was introduced and all the phases of the production were explained. At the end there were the writer's thoughts about making the game and the problems he encountered.

NGUI's usefulness for mobile games and apps was seen during the making of the game. It saves time because of its graphic setting that also allows replacing pictures with newer ones. The company can easily use this method to make the game to look the way they want with their own graphics.

SISÄLLYS

KESKEISET TERMIT JA LYHENTEET

Unity	Pelimoottori, jolla opinnäytetyön peli tehtiin.
Android	Taulutietokone. Laite, jolle peli ohjelmoitiin.
JavaScript	Ohjelmointikieli, jota voi käyttää Unityssä. Toiselta nimeltään UnityScript.
C#	Ohjelmointikieli, jota voi käyttää Unityssä.
Rigidbody	Jäykkäkappale, joka antaa Unityn valmiiksi määrittelemät ominaisuudet, kuten painovoiman halutulle peliobjektille.
Sprite	Kuva, joka piirretään ruudulle.
Prefab	Valmisolio. Kierrätettävä objekti, jonka ominaisuudet on määritetty valmiiksi.
Blender	3D-mallinnusohjelma, jolla tehtiin pelin hahmot.
GUI	Graphical User Interface eli graafinen käyttöliittymä.
Frame	FPS:n (frames per second) eli kehysnopeuden mittayksikkö. Kehysnopeus vaihtelee laitteen suorituksesta riippuen, mutta tasapainotettuna tämä peli näyttää 30 framea sekunnissa.
NGUI	Unityn liitännäinen, jolla voi tehdä helposti graafisia komponentteja.
Tagi	Peliobjektiin lisättävä mahdollinen tunniste.

1 JOHDANTO

Sain aiheen opinnäytetyölleni paikalliselta peliyritykseltä nimeltä Kotka Games. Yritys perustettiin virallisesti vuoden 2013 kesällä (1.). Yrityksen toimipaikka sijaitsi työntekohetkellä Karhulassa olevassa mediatehdas Dakarin tiloissa. Yritys muutti uuteen toimitilaan tammikuussa 2014. Henkilökuntaa yrityksellä on vain muutama henkilö, mutta he saavat tarvittaessa vahvistusta Dakarin muilta työntekijöiltä.

Dakar on kotimainen tuotantoyhtiö, joka tuottaa ja kehittää sarjoja, elokuvia sekä mainoksia TV-kanaville niin Suomessa kuin ulkomaillakin. Dakarin tuottamia sarjoja ja elokuvia ovat esimerkiksi Virittäjät, Leijonasydän sekä Kakara (2.).

Kotka Gamesilla on tällä hetkellä tekeillä heidän ensimmäinen pelinsä nimeltä Swing Game. He myös lanseerasivat kesällä Kotkan Meripäivien aikaan samannimisen urheilutapahtuman, jossa ammattiurheilijat hyppäsivät keinusta mereen. Dakar tuottaa dokumenttia Swing Gamesta (3.).

Työtä valvoi yrityksen operatiivinen johtaja Samuli Suikkanen. Hän toimii myös web-sivuihin ja -sovelluksiin keskittyvän Heliwood-yrityksen johtajana.

Kotka Games antoi käyttööni tietokoneen työtä varten. Heidän keskittyessään omaan peliinsä, sain vapaat kädet tämän työn suunnittelemiseen. Tarkoituksena oli tehdä PC-peli ja kehittää siitä myös Android-versio. Sen tekeminen aloitettiin elokuun puolessa välissä. Aihe hyväksyttiin kuitenkin vasta syyskuussa koulujen alkaessa.

Peli valmistui parin kuukauden sisällä aloittamisesta. Kyseessä ei ole kuitenkaan täysin valmis peli, vaan siitä puuttuvat grafiikka ja äänet lähes kokonaan. Grafiikkaa pelistä löytyy kuitenkin sen verran, mitä onnistuin itse sitä tekemään tietämättä sen enempää 3D-mallinnuksesta.

Pelin tekeminen toteutettiin Unity3D-pelimoottorilla. Tässä dokumentissa tarkastellaan aluksi Unity3D:n rakennetta. Sen jälkeen kerrotaan työssä käytetystä Unityn liitännäisestä NGUI:sta, joka täytyy hankkia erikseen Unity Storesta. Sitten kerrotaan lyhyesti oleellisimmat asiat pelistä, jonka jälkeen on vuorossa tekijän mietteitä tehdystä työstä ja sen vaikutuksista oman osaamisen kehittymiseen.

2 UNITY

2.1 Rakenne

Unity, tai viralliselta nimeltään Unity3d, on pelimoottori, jolla voi tehdä videopelejä eri alustoille (4.). Opinnäytetyönä tehty peli on tehty sekä PC:lle että Android-laitteille.

2.1.1 Scene

Scene on pelialue, jossa on objekteja. Sitä voisi ajatella nimensä mukaisesti näyttämönä, jossa on omat rekvisiittansa jokaisen esityksen kohtaukselle. Samoja asioita voidaan käyttää uudelleen myöhäisemmissä sceneissä, mikäli niistä on tehty valmisolio eli prefab. Scene voi myös olla tyhjä, mutta silloin peli olisi turha, koska käyttäjä ei voisi olla vuorovaikutuksessa mitenkään sen kanssa (5.).

Tässä pelissä jokainen kenttä on oma scenensä. Myös kenttien välissä olevat läpäisy- ja tarinaruudut ovat omia scenejään, joiden sisältö muuttuu koodin avulla kulloiseenkin tilanteeseen sopivaksi. Lisäksi pelin alku- ja muut valikot ovat itsenäisiä scenejä.

2.1.2 Peliobjektit

Scenen sisältö muodostuu peliobjekteista (GameObjects). Ne voivat olla esimerkiksi kuutioita, kameroita, olemattomia objekteja tai partikkeliefektejä. Ne voivat pitää sisällään koodeja, valoja tai muita efektejä ja ominaisuuksia. Peliobjektit ovat tärkeimpiä komponentteja Unityn käytössä (6.).

Peliobjekteilla on monenlaisia käyttötarkoituksia. Peliobjektit voivat myös hallita toisia objekteja koodien avulla. Niillä voi myös pitää kirjaa tapahtumista, kuten kerättyjen esineiden määrästä tai elämistä. Uusi peliobjekti luodaan työkalun GameObject-valikosta valitsemalla haluttu muoto. Sieltä voi myös luoda tyhjän. Tämän jälkeen peliobjekti ilmestyy scenelle ja sille voi antaa ominaisuuksia. Peliobjektien perusominaisuuksia ovat koko, kääntyvyyssuunta sekä sijoittuminen scenellä.

Peliobjektia ei voi siirtää sceneltä toiselle suoraan. Jos on esimerkiksi tehty pelattava hahmo ja sitä tahdottaisiin käyttää seuraavassa kentässä, on luotava prefab eli valmisolio. Sellainen luodaan valitsemalla valikosta Assets->Create->Prefab. Tämä luo tyhjän valmisolion projektin kansioon. Sen jälkeen vain tarvitsee vetää haluttu peliobjekti sceneltä tai hierarkiavalikosta valmisolioon, jonka jälkeen se on valmis käytettäväksi muilla sceneillä. On kuitenkin muistettava, että jos siihen tekee scenellä muutoksia, ne eivät siirry automaattisesti muilla sceneillä olevien valmisoliosta luotujen kloonien ominaisuuksiksi. Tiedot täytyy hyväksyä painamalla Apply-nappia muutetun peliobjektin ominaisuuksista. Tämän jälkeen tiedot päivittyvät kaikkiin sceneihin. Tietojen muuttumattomuutta voi myös käyttää hyödyksi tekemällä esimerkiksi vihollisia, jotka liikkuvat eri nopeuksilla (7.).

2.1.3 Fysiikka

Unityn fysiikat pitävät sisällään törmäysmaskit sekä painovoimat. Painovoiman saa peliobjektille lisäämällä sille rigidbody-ominaisuuden (8.). Painovoima on Unityn sisäisesti määritetty ominaisuus. Rigidbodyn saa lisättyä valittuun peliobjektiin valitsemalla valikosta Component->Physics->Rigidbody.

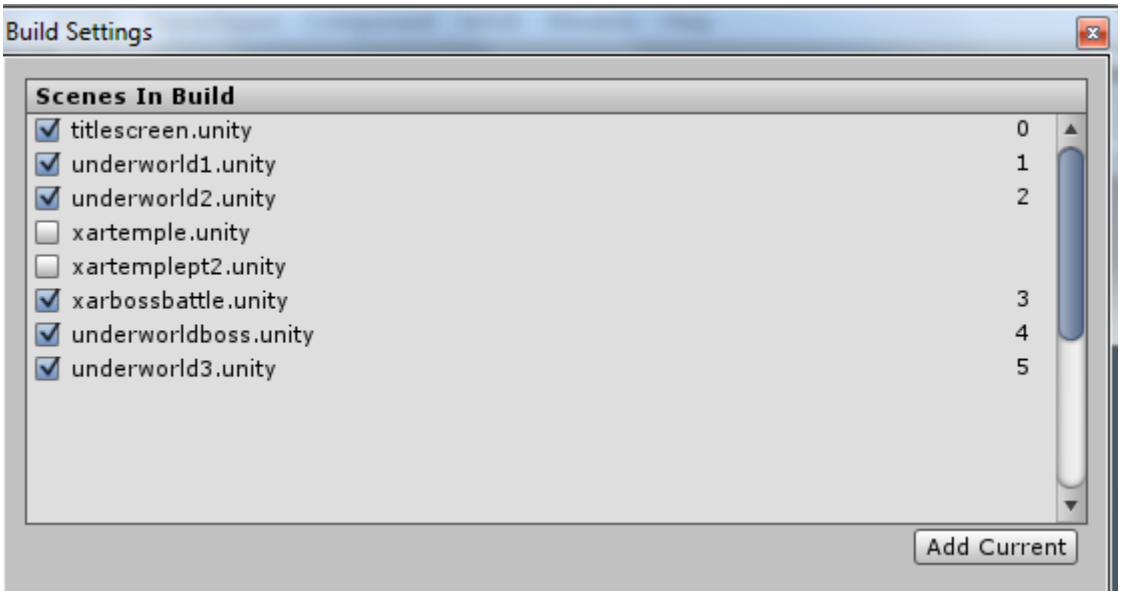
Törmäysmaskit ovat alueita, joita hyödynnetään koodissa. Niillä määritetään erilaisia toimenpiteitä, joita tapahtuu kahden kappaleen törmätessä, kappaleen ollessa jonkin alueen sisällä tai sen lähtiessä pois alueelta. Maskeja on erilaisia: laatikko, sylinteri, pallo, maasto, pyörä sekä mesh (8.). Mesh on mikä tahansa muoto, jonka 3d-malli löytyy pelin resursseista. Esimerkiksi pelissäkin löytyvällä katulampulla on itsensä muotoinen alue, jolla se ottaa huomioon törmäykset. Mesh-maskin huono puoli on kuitenkin se, että jos kappaleella on jossain ohut kohta, toinen kappale saattaa mennä siitä läpi ilman törmäystä.

2.1.4 Koodaus

Unityllä koodaaminen onnistuu valitsemalla joko Assets->Create->Haluttu koodikieli tai vaihtoehtoisesti valitsemalla peliobjekti ja painamalla sen Add Component –nappia ja valitsemalla New Script->Haluttu koodikieli sekä antamalla luotavalle koodille nimi, ennen kuin painaa Create and Add –nappia. Ilman nimen antamista koodi saa vakionimen NewBehaviourScript. Unity ymmärtää kolmea koodikieltä, jotka ovat Javascript, C# ja Boo (9.).

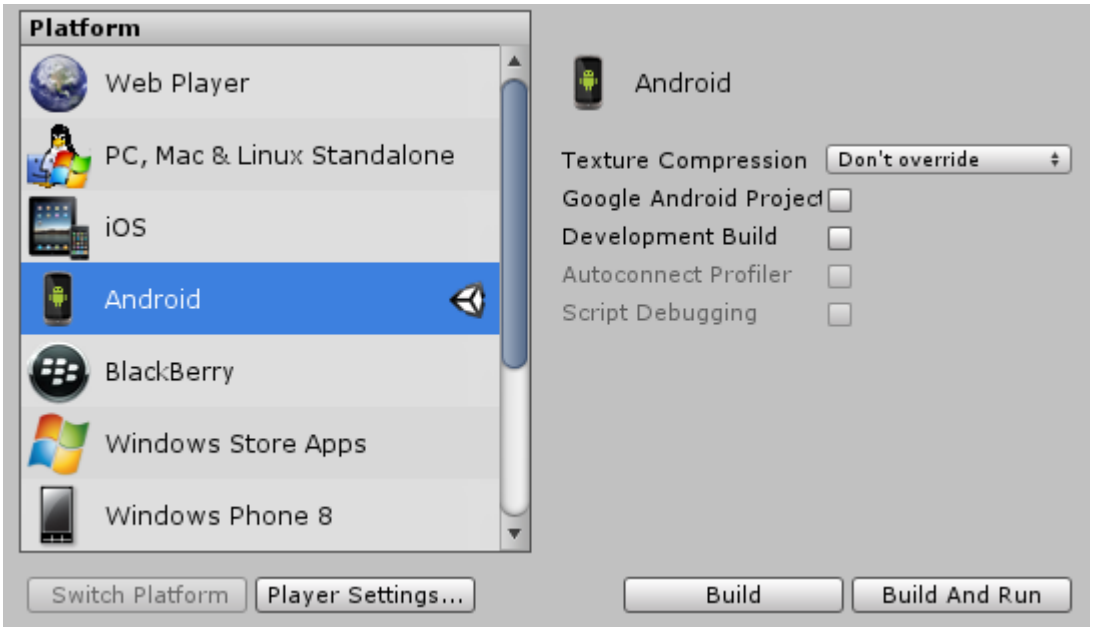
2.1.5 Kasaus

Kasaus- eli Build-vaiheessa Unity kääntää ja rakentaa pelin valitulle alustalle. Scenet lisätään peliin manuaalisesti valitsemalla File->Build Settings, joka avaa uuden ikkunan. Tässä ikkunassa on paneeli, jonka otsikkona on ”Scenes In Build”. Sen alapuolella on ”Add Current” -nappi. Painamalla sitä sillä hetkellä aktiivisen scenen nimi ilmestyy paneeliin checkboxin kanssa. Checkboxin ollessa valittuna scene on mukana valmiissa pelissä. Jos scenejä on useampia, niiden paikkaa voi vaihdella raahaamalla. Saman scenen voi kierrättää lisäämällä sen uudestaan listaan. Tätä tapaa käytettiin tehtäessä väliruudut kenttien välille.



Kuva 1. Kasauksen scenejen paneeli.

Kääntäminen halutulle alustalle tapahtuu valitsemalla sellainen kuvan 1 esittämän näkymän alapuolella olevasta Platform-laatikosta (kuva 2). Sen jälkeen painetaan ”Switch Platform” -nappia, jonka jälkeen Unity kääntää kaikki projektin tiedostot oikeanlaiseksi. Tässä tapauksessa valinta oli Android. Osa koodeista ei välttämättä käänny oikein, mikä johtaa virheilmoituksiin.



Kuva 2. Alustan valitseminen.

Kun virheet on korjattu ja yritetään painaa Build-nappia, Unity valittaa Android SDK:n eli software development kitin määrityksen puuttumisen. SDK:n määrittämiseksi täytyy valita Unityn valikoista Edit->Preferences, joka avaa uuden ikkunan. Tästä ikkunasta valitaan External Tools, jonka jälkeen voidaan määrittää Android SDK Location. Kyseinen SDK on täytynyt ladata netistä Androidin kehittäjäsivuilta. Kun Android SDK on asennettu ja Unityyn määritelty SDK:n sijainti, voidaan painaa Build-ikkunassa olevaa Build-nappia, jolloin pelistä luodaan APK-tiedosto. Tiedoston voi siirtää Android-laitteeseen. Painamalla kuvassa 2 näkyvää ”Build and Run” –nappia Unity kasaa pelin ja käynnistää sen itse laitteessa saman tien.

2.2 Keskeiset metodit

Työssä on käytetty suurimmaksi osaksi Unityn valmiita metodeja. Näitä metodeja ovat peliobjektin luontitilanteessa käytettävä Start, jokaisen framen aikana toimiva Update, sekä törmäystarkasteluita varten käytettävät OnTrigger- ja OnCollision (9.).

Start-metodia käytettiin yleensä etsimään peliobjektin tarvitsemia muita objekteja scenellä. Sitä käytettiin myös aloittamaan joissakin tapauksissa Coroutine. Coroutine on aikaan liittyvä metodi, jota voidaan toistaa haluttaessa. Se käynnistetään kirjoittamalla ”StartCoroutine(nimi());”. Coroutineja käytettiin tietyissä ammuksissa, jotka räjähtävät vähän ajan kuluessa niiden laukaisusta. Coroutineihin saadaan

aikaviive asettamalla ”yield return new WaitForSeconds(viive);” haluttuun kohtaan (10.).

Update-metodi käynnistyy jokaisella pelisilmukan kierroksella useita kymmeniä kertoja sekunnissa (9.). Se toistaa kaiken uudelleen, jos siihen ei aseteta rajoitetta, esimerkiksi boolean-arvolla. Pelissä Updatea käytetään pelaajan mittarien elpymisessä.

Updaten käyttö on raskasta, mikä näkyy Android-laitteella. Siksi on järkevämpää käyttää sitä vain välttämättömissä tilanteissa. Suurimman osan Update-metodeista voi helposti korvata Coroutinella, joka toistuu esimerkiksi 0,5 sekunnin välein.

OnTrigger- ja OnCollision-metodeilla on kolme muotoa: Enter, Stay sekä Exit. Enter tapahtuu silloin, kun kahden peliobjektin törmäyslaatikot kohtaavat. Stay tapahtuu joka framen aikana, jona objektit pysyvät kosketuksissa. Exit puolestaan toteutuu niiden erotessa (11.).

Erona OnTriggerillä ja OnCollisionilla on se, että Trigger vaatii toiselle objekteista IsTrigger-valinnan päälle törmäyslaatikosta. Trigger-objektit menevät myös muista objekteista läpi, eivätkä ne vaikuta muiden objektien painovoimaan törmäyksessä. Triggeriä käyttävät pelin ammukset sekä tasojen loput.

3 NGUI

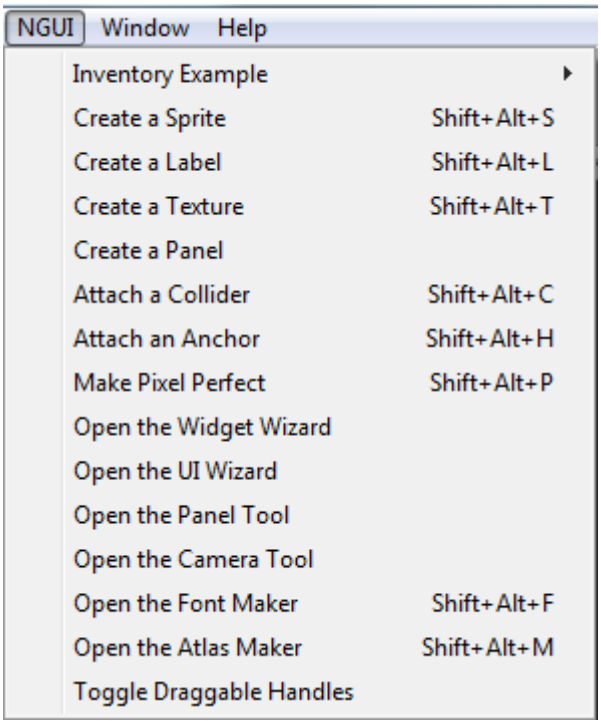
3.1 Yleistä

NGUI on Tasharen Entertainmentin luoma liitännäinen Unityyn. Hinnaltaan se on 95 dollaria.

NGUI sisältää lisäkoodeja, jotka helpottavat Unityllä työskentelyä. Koodeja voi käyttää sellaisenaan tai niitä voi muokata haluamakseen. NGUI:n päällimmäinen tarkoitus on tarjota toimiva UI-järjestelmä, jolla on vakaa tapahtumankäsittely.

NGUI:lla pystyy tekemään helposti komponentteja, kuten nappeja, kuvia tai jopa fontteja (kuva 3). Komponenttien ominaisuuksien muokkaaminen onnistuu helposti muuttamalla komponentin ominaisuuksia ominaisuusikkunassa. Ominaisuuksien

oletusarvot toimivat sellaisenaankin. Parasta NGUI:n käytössä on, ettei Unityä tarvitse laittaa Play-tilaan, vaan asetetut muutokset näkyvät suoraan NGUI:n ikkunassa (12.).



Kuva 3. NGUI-valikko.

3.2 Komponentit

3.2.1 UI

NGUI:n tärkein komponentti on UI-komponentti. Mikään NGUI:n komponentti ei toimi, jollei se ole UI:n sisällä. Tästä syystä aloitettaessa tekemään mitä tahansa NGUI:lla, se luo UI:n ja muut tarvittavat objektit, vaikka tekijä olisi painanut napin luontia. Nämä muut objektit ovat UI-kamera, ankkuri sekä paneeli. UI:n koon saa määritellä itse antamalla sille minimi- ja maksimikorkeudet. UI skaalaa itsensä sen mukaan, minkä kokoinen näyttö on (13.).

Samalla scenellä voi olla useampi UI samanaikaisesti. Silloin on kuitenkin otettava huomioon, miten sijoittaa kuvat, sillä ne voivat peittää toisensa. Tapausta voi miettiä oikean maailman kannalta: Jos sinulla on kasa leikeltyjä lappuja ja sommittelet niitä tietylle alueelle, niiden mennessä päällekkäin ei näe, mitä toisen alapuolella on.

UI:n luontivaiheessa on valittava, millä kerroksella se sijaitsee ja millaista kameraa se käyttää. Kamera ei ole pakollinen tai se voi olla myös joko 2D- tai 3D-kamera. Jos

kameraa ei ole, UI on kuin peliobjekti, jonka sijainti on sama, kuin mihin tekijä on sen laittanut. 2D-kamera kuvaa kaksiulotteisesti ja 3D kolmiulotteisesti. Pelin valikoihin ja näkymiin käytettiin 2D-kameraa.

Kameran ansiosta UI:n voi sijoittaa minne tahansa scenellä (kuva 4). Kameraa käytettäessä on kuitenkin hyvä tarkistaa, että se on määritetty kuvaamaan oikeaa kerrosta, sillä muuten UI ei näy. Kamera käyttää NGUI:n omaa kamerakoodia, jonka ominaisuuksia pystyy säätämään.

Kuva 4. Kuvassa näkyvä pyöristetty neliö teksteineen on osa UI:ta. Samoin sen yläpuolella olevat himmeä palkki, ikoni ja tekstit kuuluvat UI:hin.

UI:ssa käytettävät ankkurit mahdollistavat objektien suhteellisen sijoittelun. Ankkuria ei voi manuaalisesti siirtää, joten se kannattaa liittää tyhjään peliobjektiin, jonka sisään laitetaan ne objektit, joiden halutaan olevan suhteessa valittuun kohtaan.

3.2.2 Paneeli

Paneeliin säilötään kaikki halutut objektit. Paneelin avulla voidaan säätää sen sisällä olevien objektien näkyvyyttä. Niistä voi tehdä läpinäkyviä muokkaamalla paneelin alpha-kanavaa.

Paneeleita voi olla UI:n sisällä myös useampia. Sen vuoksi voi esimerkiksi tehdä osittain läpinäkyvän taustan yhdellä paneelilla ja laittaa toisen paneelin sen päälle.

Paneeleita voi myös piilottaa. Esimerkiksi pelissä oleva taukoruutu (kuva 5) on kokoajan olemassa, mutta se ei näy ennen kuin pelaaja haluaa tauon.

Kuva 5. Peli on tauotettu.

3.2.3 Vekottimet

Vekottimet eli widgets ovat NGUI:n nimitys napeille sekä muille valmiille komponenteille, joita voi käyttää sellaisenaan tai niitä voi muokata. Niitä hallitaan Widget Wizardilla.

Widget Wizardilla valitaan haluttu vekotin. Se voi olla nappi, kuvanappi, valintaruutu, latauspalkki, liukusäädin, input-kenttä, scrollauspalkki, popup-lista tai popup-menu. Vekottimet sisältävät myös toiminnallisuudet. Esimerkiksi liukusäätimessä on nappi, jota voi vetää, jolloin palkki liikkuu sen mukaan, mihin napin asettaa. Liittämällä koodin äänienkuuntelijaobjektiin, voidaan säätää äänenvoimakkuutta ottamalla arvot suoraan liukusäätimestä. Tässä pelissä liukusäädintä ei kuitenkaan ole käytetty.

Vekottimissa on myös paljon koodeja, jotka eivät ole välttämättömiä. Esimerkiksi napissa on erikseen koodi, joka suurentaa nappia hiiren ollessa sen yläpuolella ja värjäten napin vihreällä hehkulla. Väriä voi vaihtaa, mutta sen voi myös poistaa kokonaan. Ohjelmoija voi tehdä myös helposti oman nappinsa luomalla pelkän kuvan, asettamalla sille osumalaatikon sekä kirjoittamalla koodin, joka pitää sisällään OnClick-metodin, joka on NGUI:n oma lisäys Unityyn. NGUI:n napit tuovat

kuitenkin sen verran valmista koodia, että niitä on helpompi soveltaa niiden selkeyden vuoksi. Lisäksi mikään ei estä poistamasta napilta siinä olevia turhia koodeja kokonaan. Pelissä käytettiin vekottimista nappeja, kuvia ja kuvanappeja. Niiden avulla sai tehtyä kaiken tarpeellisen.

3.2.4 Fontti

NGUI:lla voi tuoda minkä tahansa valmiin fontin peliin tai jopa tehdä itse oman fontin. Fontit voivat olla tekstitiedostoon määritettyjä tai bittikarttaan (kuva 6) piirrettyjä merkkejä.

Kuva 6. Esimerkki bittikarttafontista.

3.2.5 Atlas

Atlas on kuvajoukko. Kaikki atlaksessa olevat kuvat ovat sitä käyttävän spriten käytössä. Haluttaessa vaihtaa spriten kuvaa toiseksi, on uuden kuvan löydyttävä

samasta atlaksesta kuin alkuperäinen. Atlaksilla on maksimikokona 4096 x 4096 pikseliä. Kuitenkin mobiililaitteilla maksimikoko on 1024 x 1024 pikseliä, sillä mobiililaitteiden graafiset ominaisuudet eivät riitä suurten atlasten käsittelyyn.

Jos poistetaan kuva projektikansiosta, se jää silti atlakseen. Kuva useimmiten myös korruptoituu siten, ettei ohjelma tunnista koodissa kuvan vaihtumista, jonka vuoksi koko atlas muuttuu käyttökelvottomaksi, jollei kuvaa poisteta atlaksesta manuaalisesti.

4 ANDROID

4.1 Historia

Android on Android Inc.:n kehittämä kosketustekniikkaa hyödyntävä mobiililaite. Android kuuluu nykyisin Googlen omistukseen (14.). Ensimmäiset Android-laitteet julkaistiin Suomessa vuonna 2009 (15.).

4.2 Ominaisuudet

Androidin perusominaisuuksiin kuuluu tiedon tallentaminen, viestintä ja verkkoselaus. Se tukee Javaa ja käyttää peruskehitysympäristönään Eclipse-ohjelmaa, jossa Javaa käytetään koodikielenä (16.).

Android-laitteet tunnistavat usean samanaikaisen kosketuksen. Tämän opinnäytetyön pelissä kyseistä ominaisuutta on käytetty pelaajan erikoiskyvyn käytössä.

Androidilla pystyy myös ajamaan useaa ohjelmaa samanaikaisesti. Android sammuttaa automaattisesti sovellukset, jotka ovat olleet pitkään joutilaana.

4.3 Android pelialustana

Android on yksi myydyimmistä mobiililaitteista. Vuonna 2013 sillä oli lähes 70 % markkinaosuudesta (17.).

Androidin malleissa on suurta hajontaa, joka ulottuu heikkotehoisista pienistä laitteista suurempikokoisiin laitteisiin, joilla on parempi suoritusteho. Sen vuoksi

ohjelmoitaessa on otettava huomioon missä menee raja, ettei vähempitehoisia laitteita jouduta sulkemaan pois julkaisusta.

Tässä tehdyssä pelissä käytettiin kahta erilaista Android-laitetta. Ensimmäinen oli Samsung Galaxy S3, jossa on 10 tuuman näyttö. Toinen laite puolestaan oli laadultaan ja kooltaan pienempi Acer Iconia B1-A71, jonka näyttö oli vain 7 tuumaa.

4.4 Rajoitukset

PC- ja Android-pelien suorituskyvyissä on eroja. Verrattaessa tämän opinnäytetyön PC- ja Android-versioita keskenään, helposti tunnistettavin ero löytyy grafiikasta. Osa johtuu NGUI:n atlaksista, sillä Android ei pysty näyttämään niitä, jos atlaksen koko ylittää 1024 x 1024 pikselin rajan. Toinen graafinen alemmuus on tekstuureissa, sillä Unity pakkaa ne eri tavalle Androidille. Tekstuureista saa halutessaan identtiset PC-version kanssa, vaikkakin suorituskyvyn kustannuksella.

Ruudunpäivitys on Androidilla huomattavasti heikompaa. Peliä tehtäessä näkyi vakavia hidastumisia ruudulla silloin, kun siinä oli paljon samanaikaisia visuaalisia tapahtumia. PC:llä vastaavassa tilanteessa ei kyseistä hidastumista tapahtunut. Tästä syystä Android versiosta on karsittu tiettyjä objekteja.

4.5 Myynti

Vaikkei tätä peliä saatu vielä myyntivaiheeseen saakka, on siitä hyvä kertoa edes jotain. Android-sovelluksia ja -pelejä myydään Googlen Play-kaupassa. Tekijä saa itse päättää sovelluksensa hinnan (18.). Tekijä voi myös rajata julkaisua alueen, laitteen varustuksen tai mallin mukaan (19.).

5 PELI

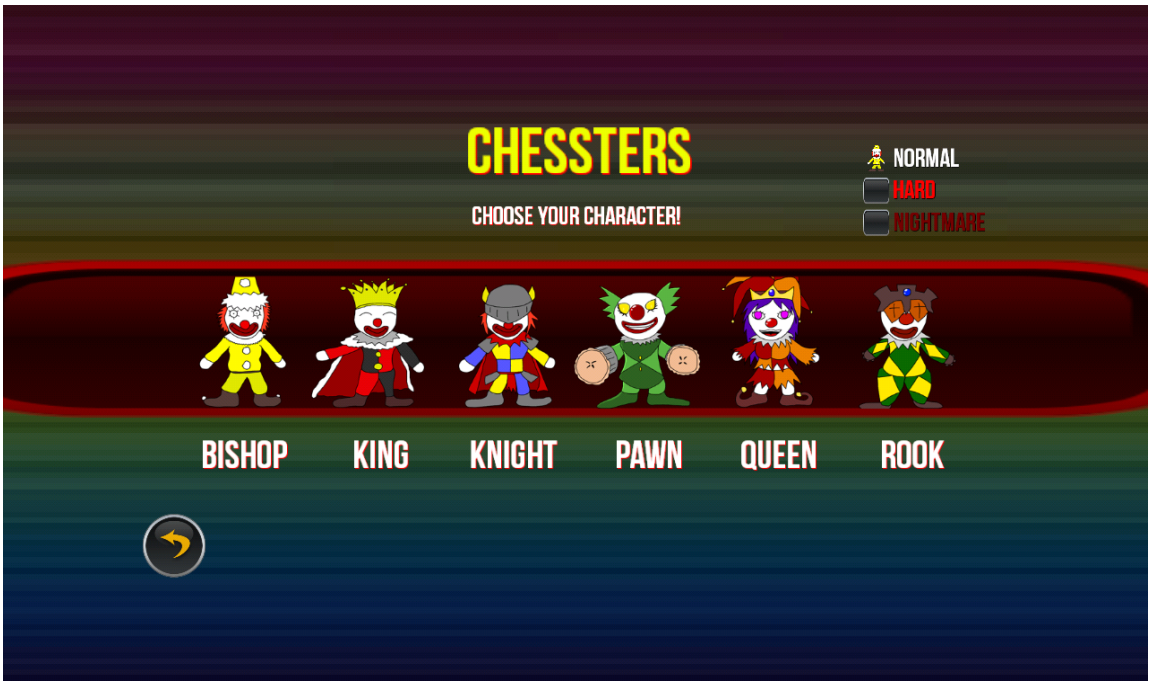
5.1 Peli-idea

Pelin ideana oli tehdä seikkailu-/tasohyppelypeli. Pelin nimi on Chessters. Pelissä kentät ovat yksinkertaisia siirtymisiä paikasta A paikkaan B. Kentissä voi olla pieniä ongelmanratkaisutehtäviä vaikeuttamassa matkaa.

Pelissä on kolme vaikeusastetta: normaali, vaikea sekä painajainen. Näiden vaikeusasteiden erot näkyvät vihollisten kestävyydessä sekä voimassa.

5.2 Hahmot

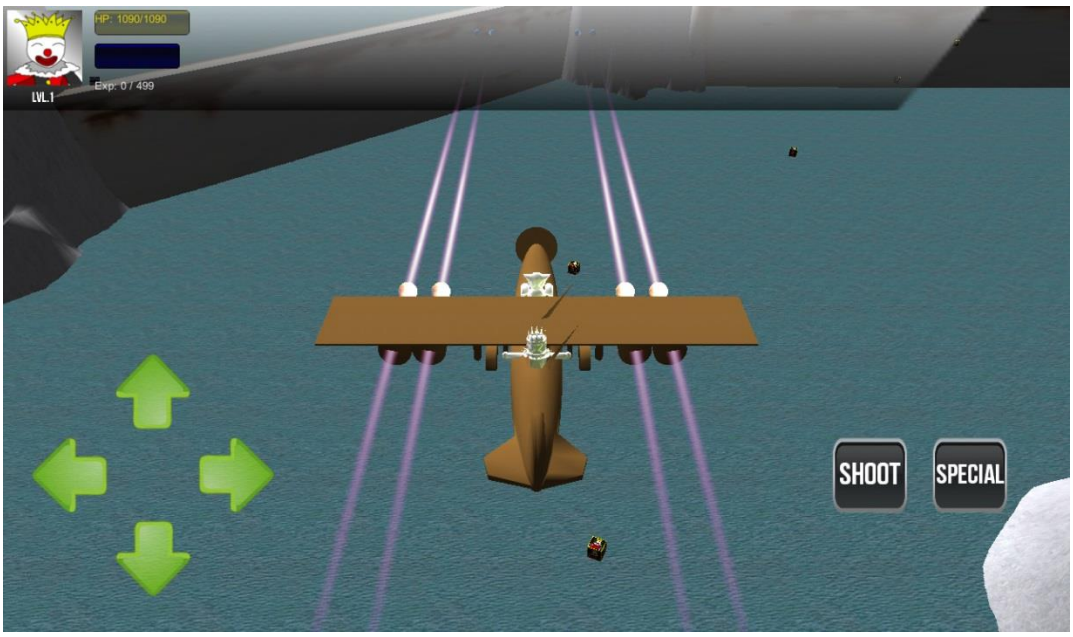
Pelaajalla on valittavanaan yhteensä kuusi eri hahmoa (kuva 7). Alussa hahmoista on käytössä vain yksi. Muut hahmot täytyy ansaita läpäisemällä peli eri vaikeusasteilla tai kehittämällä yhtä hahmoa tarpeeksi. Hahmojen 3D-mallit on mallinnettu Blender-ohjelmalla.



Kuva 7. Hahmot pelin hahmonvalintaruudussa.

Androidilla pelattaessa hahmot liikkuvat koskettamalla ruudulla haluttua suuntaa. Hyppääminen onnistuu koskettamalla itse hahmoa. Ampuminen onnistuu painamalla vihollista sormella. Erikoiskykyä puolestaan käytetään koskemalla kolmella sormella ruutua. PC-versiossa kaikki toiminta tehdään näppäimistöllä.

Hahmoilla on myös käytössään ajoneuvoja, kuten auto ja lentokone. Niitä käytetään vain tietyissä kentissä. Niiden hallitsemiseen käytetään ruudulla olevia nappeja (kuva 8).



Kuva 8. Ruutukaappaus lentokoneesta toiminnassa.

5.3 Viholliset

Pelissä on kolmen tyyppisiä perusvihollisia: hitaat, keskiverrot ja nopeat. Jokaisesta tyypistä on olemassa liikkuvat yksilöt sekä paikallaan pysyvät. Vihollisten ampumisnopeus on kääntäen verrannollinen vihollisen havainnointialueeseen.

Perusvihollisten lisäksi pelistä löytyy pomovastustajia. Ne ampuvat useamman ammuksen kerrallaan ja ovat huomattavasti kestävämpiä kuin tavalliset viholliset.

5.4 Hahmonkehitys

Pelin syvyyttä lisää hahmonkehityssysteemi. Kullakin hahmolla on oma kokemustasonsa, jonka nousemiseksi on tuhottava vihollisia tai kerättävä kentistä löytyviä kokemuslaatikoita. Kasvamiseen tarvittava määrä nousee jokaisen tason myötä matemaattisella kaavalla 450 * nykyinen taso.

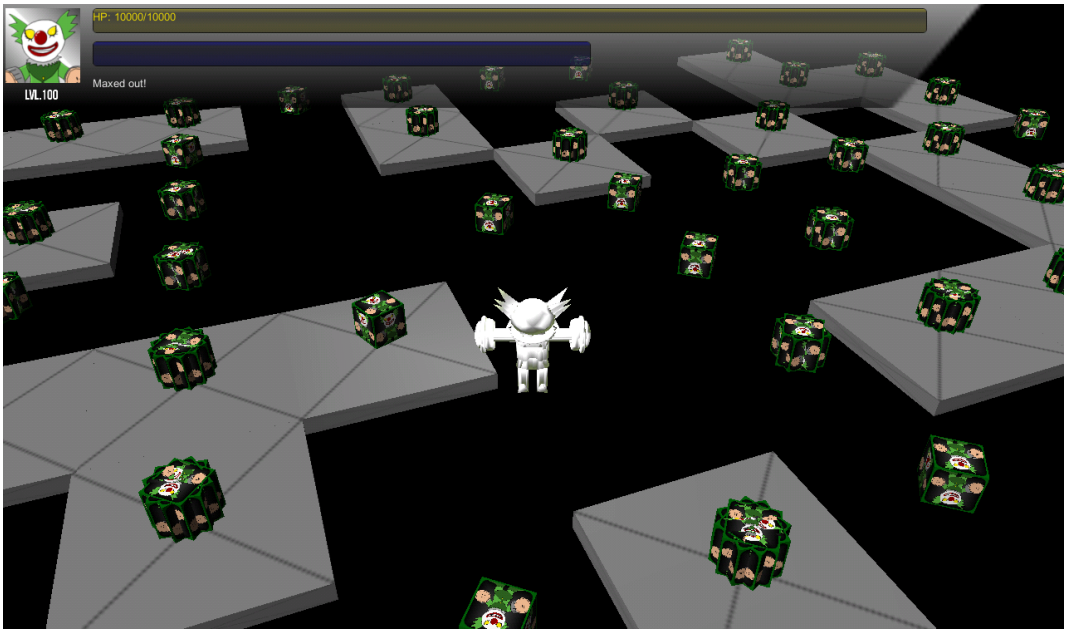
Uusien kokemustasojen myötä hahmon elämämittari kasvaa, ja joka kymmenes taso tarjoaa uusia kykyjä hahmolle (kuva 9). Jokaisella hahmolla on omat kykynsä.

Kokemustaso	Kyky
10	Enemmän ammuksia
20	Kehittyneet ammukset
30	Passiivinen erikoiskyky
40	Kykymittarin kasvu
50	Liikkumisnopeuden kasvu
60	Kehittyneet ammukset
70	Vahingon puolitus
80	Hyppykorkeuden kasvu
90	Automaattinen parantuminen
100	Kykymittarin palautumisen kasvu

Kuva 9. Taulukko hahmonkehityksestä saatavista kyvyistä.

5.5 Minipeli

Peli pitää sisällään minipelin (kuva 10), jonka avulla pelaaja voi kehittää hahmojaan nopeammin. Minipelissä kerätään pieniä laatikoita, joista saa kokemusta. Laatikot ovat laattojen päällä. Muutaman sekunnin välein kyseiset laatat joko luovat uudet laatikon tai tuhoavat itsensä.



Kuva 10. Kuvakaappaus minipelistä.

6 TYÖVAIHEET

6.1 Alkuvaihe

Työn ensimmäinen vaihe oli luoda liikuteltava hahmo sceneen. Perusominaisuudet, kuten nopeus, jolla hahmo liikkuu eteen ja taaksepäin, hyppääminen sekä kääntyminen, olivat ominaisuudet, jotka hahmolle annettiin. Seuraavana oli vuorossa kameran koodi, jotta se seuraisi, mihin hahmo liikkuu.

Kun hahmo ja kamera oli saatu tehtyä, oli vuorossa lisätä sceneen esteitä seiniksi ja tasanteiksi, joille voisi hypätä. Nämä olivat suoraan Unityn omia muotoja.

Seuraavaksi kenttään lisättiin vihollinen, joka ampuisi pelaajaa tämän ollessa lähellä. Koodiin määritettiin ampumiselle aikaväli. Sen avulla luotiin kolme eri tahtiin ampuvaa vihollista.

Vihollisten tuhoamiseksi pelaajalle annettiin myös kyky ampua. Sekä pelaaja että viholliset saivat elämänpisteet, joiden tyhjentyessä hahmo tuhoutuisi.

6.2 Hahmojen luonti

Alkuvaiheen jälkeen ideoitiin, mitä peli pitäisi sisällään. Hahmojen ulkonääöistä tehtiin vedokset ja niistä luotiin 3d-mallit Blender-mallinnusohjelmalla. Unityn puolella jokaiselle pelattavista hahmoista tehtiin omat ampumiskoodinsa sekä annettiin omanlainen erikoiskyky.

6.3 Kenttien tekeminen

Kenttiä tehtiin yhteensä 12 kappaletta. Jokaiseen kenttään tuli jokin tarkoitus. Alkupään kentät oli tarkoitettu antamaan pelaajalle ote pelistä. Myöhemmissä kentissä pelaaja pääsee tutustumaan kulkuneuvoihin, kuten lentokoneeseen ja autoon (kuva 11).

Kuva 11. Ruudunkaappaus kentästä, jossa on auto.

Peliin tehtiin myös kolme pomotaistelutasoa, joissa pelaaja joutuu kohtaamaan voimakkaan vihollisen. Pomotaistelujen tarkoituksena oli antaa haastetta pelaajalle. Haasteellisuuden testaamiseksi järjestettiin muutaman hengen testaus, josta saadun palautteen perusteella peliin toivottiin helpotusta ja saavutuksen tunnetta. Vastauksena palautteeseen peliin tehtiin hahmonkehityssysteemi, jotta pelin pelaaminen helpottuisi mitä enemmän sitä pelaisi.

6.4 Valikot

Aluksi peliin tehtiin alku- ja hahmonvalintaruudut. Alkuruudussa on vaihtoehtoina käydä katsomassa pelin kontrollit tai aloittaa peli (kuva 12). Aloittamalla pelin siirrytään hahmonvalintaruutuun, jossa pelaaja valitsee hahmonsa koskettamalla haluttua hahmoa. Sen jälkeen siirrytään kentänvalintaan, joka toimii myös kosketuksella. Tämän jälkeen itse peli alkaa, jolloin kenttään luodaan valittu hahmo. Valittu hahmo saadaan tiedosta, joka tallennetaan hahmonvalinnanyhteydessä muistiin.

Kuva 12. Alkuruutu.

Jokaisen kentän jälkeen ilmestyy voittoruutu, jossa on kuva pelatusta hahmosta ja teksti ”Level Complete” (kuva 13). Kuva vaihtuu kyseiseksi hahmoksi katsomalla samasta tallennetusta tiedosta, millä kenttään luodaan hahmo.

Kuva 13. Kentänläpäisyruutu.

Pelin lopussa on lopputekstiruutu, jossa kamera liikkuu alaspäin ja näyttää pelin tekijät ja testaajat. Se kestää puolitoista minuuttia, ennen kuin palaa takaisin alkuvalikkoon. Pelaajan kuollessa kentissä ruutuun ilmestyy laatikko, joka kysyy

halutaanko jatkaa. Vastaamalla myönteisesti kenttä alkaa alusta. Kieltäytymällä siirrytään pelin Game Over -ruutuun (kuva 14), josta palataan lopulta pelin alkuvalikkoon.

Kuva 14. ”Game Over” -ruutu.

6.5 Android-versio

Valikoiden tekemisen jälkeen peli siirrettiin Androidille. Kääntäminen ei toiminut sellaisenaan, sillä pelin atlakset olivat liian suuria. Tämä johti niitä käyttävien tekstuurien puuttumiseen. Ongelma korjaantui skaalaamalla tekstuurit pienemmiksi.

Päästessässään pelaamaan kenttiä, hahmo ei pystynyt liikkumaan eikä hyppimään, sillä PC-versiossa liikkuminen tapahtui nuolinäppäimillä ja hyppääminen välilyöntiä painamalla. Ampuminen, erikoiskyvyn käyttö ja paussi toimivat, mutta eivät kuitenkaan järkevällä tavalla, sillä Unityssä olin määrittänyt nämä kolme toimintoa toimimaan projektin Input-asetuksissa näppäimien lisäksi myös hiiren painikkeilla. Vasen klikkaus ampui, oikea käytti erikoiskykyä ja kolmosnäppäin laittoi pelin paussille. Androidilla yksi sormikosketus ampui, kaksi sormea käytti erikoiskykyä ja ampui, sekä kolme sormea tai useampi teki kaikki kolme asiaa.

Kierrättämällä suurinta osaa koodista, peliruudun alalaitaan luotiin NGUI:lla nuolinäppäimet sekä napit muille toiminnoille. Napit veivät testaajien mielestä liikaa tilaa. Ehdotuksena oli tehdä pelistä napiton.

Uusi tapa liikkumiseen tapahtui sädetyksen (Raycast) avulla. Painettaessa kohtaa ruudulla sen koordinaatit otetaan talteen. Hahmo kääntyy ja liikkuu painettua kohtaa kohti, kunnes pelaaja päästää irti laitteesta. Muiden ominaisuuksien toimimiseksi täytyi tehdä poikkeuslauseita, joissa verrattiin painetun peliobjektin tagia. Jos kyseessä oli pelaajaksi määritetty, hahmo hyppäisi. Jos se olisi vihollinen, hahmo ampuisi. Erikoiskyvyn käyttämiseksi päädyttiin kolmen sormen samanaikaiseen kosketukseen.

Verrattuna PC-versioon, Androidin suorituskyky oli hidasta tietyissä kohdissa. Suorituskykyyn vaikutti liian moni tapahtuva asia. Hidastumisen hillitsemiseksi pelistä täytyi poistaa liialliset partikkeliefektit ja muutama koristeiksi tarkoitetut monimutkaiset 3d-mallit.

Pienemmillä Android-laitteilla pelin ohjaaminen oli vaikeaa, sillä kamera oli liian kaukana. Tämä teki hyppäämisestä vaikeaa, sillä hahmo oli juuri ja juuri pelaajan sormen kokoinen, jonka vuoksi peli saattoi rekisteröidä kosketuksen maahan eikä hahmoon. Ongelma ratkesi tekemällä kameralle laitteen ruudunleveyden tarkistuksen. Leveyden ollessa pieni, kamera olisi lähempänä.

6.6 Hienosäätö ja viimeistely

Viimeiset korjaukset olivat käyttäjäpalautteesta saatuja ehdotuksia. Suurin näistä oli hahmon kääntymisen ärhäkkyys. Koskettaessa ruutua, hahmo pyörähti välittömästi painettuun suuntaan. Jos kosketus ei ollut täysin hahmosta katsoen suoraa ylöspäin ruudulla, liikkuisi hahmo kaarevasti. Onneksi Unityssä oli yksinkertainen ratkaisu: Slerp. Slerp antaa tasaisen viiveen kääntymiseen, jonka vuoksi hahmoa oli helpompi kontrolloida.

Toinen testaajia harmittanut asia oli seinät, jotka etenkin pienemmässä laitteessa peittivät näkyvyyden niiden ollessa hahmon ja kameran välissä. Se ratkesi muuttamalla seinät väliaikaisesti läpinäkyviksi (kuva 15). Tämä muutos tapahtuu siten, että kamera laskee etäisyyden ja antaa näille objekteille oman koodin, joka hoitaa läpinäkyväksi muuttumisen. Tähän koodiin täytyi lisätä kuitenkin tarkistus, ettei myös pelaajan hahmo tai ajoneuvot muuttuisi läpinäkyväksi. Se onnistui tagien avulla.

Kuva 15. Läpinäkyvyysefekti pelissä.

Kenttiin lisättiin viimeistelyvaiheessa myös uusia esteitä, kuten ovia, joita täytyi ampua, jotta ne siirtyisivät syrjään. Lisäksi peliin tehtiin minipeli, jossa hahmoja voi kehittää nopeasti.

Lopuksi järjestettiin viimeinen testaus. Kaikki toimi suunnitelmien mukaan. Jokaisen peliä testanneen henkilön nimi löytyy pelin lopputeksteistä.

7 POHDINTA

7.1 Oma kehittyminen

Tämän opinnäytetyön tekemisen aikana opin Unitystä paljon asioita jo pelkästään kokeilemalla. Kaiken tarpeellisen oppiminen, kuten peliobjektien käyttäminen halutulla tavalla, vei oman aikansa, mutta mikään ei tuntunut mahdottomalta, kun asioita tarkasteli lähemmin.

Ongelmia kohdatessani opin etsimään ratkaisuja Unityn manuaalista. Manuaalin käyttäminen vei aluksi aikansa, mutta loppuvaiheessa sitä tuli käytettyä enimmäkseen virkistämään muistia. Muissa ongelmissa, kuten tekstuurien puuttumisesta atlaksesta Android-versiossa, käytin ratkaisemiseen asiaankuuluvia foorumeja, tässä mainitussa tapauksessa NGUI:n kotisivujen omaa keskustelupalstaa.

Ohjelmointi ei loppujenlopuksi ole muuta kuin asioiden ymmärtämistä. Kun pystyy ajattelemaan jonkin asian loogisesti, pitäisi sen olla myös mahdollista toteuttaa ohjelmallisesti.

Opin ohjelmoimisen lisäksi myös parantamaan vuorovaikutustaitojani. Testaajien kanssa käydyt keskustelut pelistä olivat kehittäviä. Palautetta saadessa pystyi käymään läpi ongelmakohdat sekä kuunnella uusia ehdotuksia ja toiveita.

7.2 Jatkosuunnitelmat

Pelin tekeminen ei vain kehittänyt minua, vaan sai kiinnostukseni peliohjelmointiin kasvamaan. Aion perehtyä Unityn käyttöön enemmän, sillä uskon sen soveltuvan riittäväksi pelimoottoriksi omiin tarkoituksiini.

Tämän pelin kohtalosta en kuitenkaan osaa varmasti sanoa. Ohjelmallisesti se on muuten valmis, mutta siitä puuttuu suurin osa grafiikasta ja äänistä, jotka myös vaatisivat omat ohjelmointinsa toimiakseen oikein. Niiden puuttuminen estää kuitenkin pelin viemisen eteenpäin.

Aion jatkaa pelien tekemistä Androidille, sillä uskon siitä olevan enemmän hyötyä kuin PC-pelien tekemisestä, sillä molemmissa pystytään tekemään samoja asioita, mutta Android tarjoaa oman lisänsä kosketusominaisuuksineen. Ennen pitkää oletan siirtyväni myös iOS-laitteisiin, mutta resurssien puutteessa en voi vielä tehdä niin.

8 LOPPUYHTEENVETO

Pelin tekeminen oli mielestäni sujuvaa. Jos kohtasin ongelman, jonka ylitse en päässyt, siirryin tekemään jotain muuta ja palasin myöhemmin asiaan, löytäessäni ratkaisun. Tiiviiden testaushetkien avulla pystyin korjaamaan virheet niiden ilmaantuessa. Käyttäjäpalaute oli myös hyödyllistä, sillä se sai motivoitua minut keksimään uusia asioita peliin, jotka tekivät siitä omalaatuisemman. Aion vastaisuudessakin tehdä käyttäjätestejä tulevaisuuden projekteissani niiden tuomien mahdollisuuksien vuoksi.

Pelin kääntäminen toiselle alustalle toi aluksi ongelmia, sillä en ollut itse käyttänyt Androidia paljoa, minkä vuoksi hyvän käyttöliittymän löytyminen tuli suurimmaksi

osaksi testaajien toiveiden mukaan, sillä heillä oli enemmän kokemusta kyseisestä laitteesta. Androidilla on oikeastaan helpompi tehdä joitakin asioita, kuten ruudun koskeminen useasta kohtaa. PC:llä ei voi koskea kuin yhteen paikkaan kerrallaan hiirellä.

Jos voisin tehdä jonkin asian toisin, se olisi alussa tekemäni turhat kentät. Alussa tein kopioita samoista kentistä, joihin vaihdoin ainoastaan pelattavan hahmon. Jos tein muutoksen yhteen kenttää, sama piti toistaa myös sen kopioissa, jotta peli tuntuisi yhtenäiseltä. Olisin voinut keskittyä vain yhteen hahmoon ja jälkeenpäin todennäköisesti kehittänyt samankaltaisen tavan, jolla hahmot luodaan kenttään tallennettujen tietojen perusteella eikä ladata hahmon omaa kenttää.

Toinen asia, jonka tekisin eri tavalla, olisi panostaminen Android-versioon heti alussa, jolloin laitteen rajoitteet löytyisivät heti, eikä minun tarvitsisi jälkeenpäin poistaa ylimääräisiä objekteja, jotka hidastavat laitetta. Tapausta on kuitenkin hankalampi käsitellä näin jälkeenpäin, sillä mitään suunnitelmia ei ollut tehty, enkä edes tiennyt tekeväni pelistä Android-versiota, ennen kuin olin saavuttanut niin paljon lyhyessä ajassa.

LÄHTEET

1. Kotka Games. Kotisivut. Saatavissa: http://kotkagames.com/ [viitattu 7.11.2013]
2. Dakar. Kotisivut. Saatavissa: http://dakar.fi/ [viitattu 7.11.2013]
3. Swing Game. Kotisivut. Saatavissa: http://www.swinggame.com/ [viitattu 7.11.2013]
4. Unity3D. Verkkosivut. Saatavissa: http://unity3d.com/unity [viitattu 7.10.2013]
5. Unity Using Scene View. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/UsingTheSceneView43.html [viitattu 17.4.2014]
6. Unity GameObjects. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/GameObjects.html [viitattu 17.4.2014]
7. Unity Prefabs. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Prefabs.html [viitattu 17.4.2014]
8. Unity Physics. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Physics.html [viitattu 17.4.2014]
9. Unity Creating And Using Scripts. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/CreatingAndUsingScripts.html [viitattu 17.4.2014]
10. Unity Coroutines. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Coroutines.html [viitattu 17.4.2014]
11. Unity Create and Destroy Objects. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/CreateDestroyObjects.html [viitattu 17.4.2014]
12. Lyashenko, M. 2011. NGUI: Next-Gen UI kit. Saatavissa: http://www.tasharen.com/?page_id=140 [viitattu 26.10.2013]
13. Lyashenko, M. 2011. NGUI: UICamera. Saatavissa: http://www.tasharen.com/?page_id=148 [viitattu 17.4.2014]

14. Renshaw, N. 2011. A Brief History of Android OS. Saatavissa: http://www.lifeofandroid.com/news_detail/a-brief-history-of-android-os/ [viitattu 17.4.2014]
15. Lehtiniitty, M. 2009. Tässä on ensimmäinen Suomeenkin saapuva Google Android -puhelin. Saatavissa: http://www.puhelinvertailu.com/uutiset/2009/08/06/tassa_on_ensimmainen_suomeenkin_saapuva_google_android_puhelin [viitattu 11.2.2014]
16. Android Developer. Verkkosivut. Saatavissa: http://developer.android.com/about/index.html [viitattu 11.2.2014]
17. Lunden, I. 2014. Kantar: Android Accounted For 70% Of Smartphone Sales In Q4, But Samsung Is Now "Under Real Pressure". Saatavissa: http://techcrunch.com/2014/01/26/kantar-android-sales-in-q4-grew-in-all-big-markets-but-leader-samsung-now-under-real-pressure/ [viitattu 17.4.2014]
18. Android Developer. Developer Console. Saatavissa: http://developer.android.com/distribute/googleplay/publish/console.html [viitattu 17.4.2014]
19. Android Developer. Localization Checklist. Saatavissa: http://developer.android.com/distribute/googleplay/publish/localizing.html [viitattu 17.4.2014]